

Fall 2008/Spring 2009 Final Report: Using Games Programming in Introductory Programming Courses

Massoud Ghyam, EdD
University of Southern California
Computer Science Department

Los Angeles, CA 92612

1-213-740-4515

mghyam@usc.edu

INTRODUCTION

This paper describes some of the changes made to the CS curriculum at USC in an attempt to reverse the decline in the number of students who are declaring CS as their major and to retain the ones who declare CS as their major. To attract and keep students interested in Computer Science, professors need to redesign lower division courses to better represent the work done by professionals in the discipline and make them more interesting. Using text-based examples to teach programming fundamentals does not excite the students about their chosen major. It was decided to modify a course and experiment with the effects of adding a gaming component to the course. The modified course was the introductory programming course, CSci101. With the generous financial support of Microsoft Research the course was changed and the new format was used in 2 added lab sections.

COURSE DESCRIPTION

CSci101, Introduction to Fundamentals of Programming is the first course in a 4-course sequence that CS majors are required to take in their first two years. It covers fundamentals of computer programming concepts, and has two components, a lecture, and a lab. The professor conducts the lecture and covers concepts related to designing solutions to problems. To demonstrate the implementation of these solutions, the C++ language is used and its syntax is taught. In the lab, students spend two hours a week with hands-on exercises related to the concepts covered in the previous week's lecture. These exercises are completed in the lab with the supervision of two teaching assistants and a grader. In addition to the lab exercises, a programming assignment is given to the students weekly and it is done at home. There is one big project assigned at the end of the semester to measure each student's understanding of the material, their ability to design a solution for a big problem, and their ability to implement that solution.

CHANGES TO THE COURSES

CSci101 was modified in an attempt to better prepare games majors' needs and to improve retention. The lecture part of the course was changed to cover object oriented design and programming early in the semester, and the lab was modified by adding a game development component to the lab activities. These changes were implemented this semester (Fall 2008) and the results were assessed at the end of the semester and we are in the process of making the necessary adjustments for spring semester. The students were divided into two groups; one group which consisted of all the students who have declared themselves Computer Science major (General CS, CS Games, or Computer Engineering Computer Science) and the other group were non-CS majors. The population in the lecture is about 150 students

(divided into 3 lectures) and they are divided into smaller groups of about 25 students for the lab sessions. The Games group used GGE (GamePipe Game Engine) to develop their game, and the other group did not have any games component.

LAB CONTENTS

In a hands-on computer lab equipped with a projector that shows the instructor's desktop screen, 18 students who are in the games track spent 13 weeks to learn programming with GGE, tutored by 2 graduate teaching assistants (TAs). Each week a lab assignment is completed that covers the fundamental knowledge in program design and C++ syntax taught in the lecture as well as some game development concepts and techniques. These labs were designed to reinforce concepts covered in the two 80 minute lecture sessions during the previous week with hands on exercises. Each lab lasts 110 minutes, which is comprised of three sessions: introduction to goals and concepts, feedback-corrective tutoring, and lab exercises. All students had the same lab content for the first 8 weeks, which concentrated on teaching C++ syntax and object oriented programming. The last 5 sessions were split into 2 groups; games and regular labs. The games group learned about games concepts and games tools. All CS majors were required to participate in the games lab which concentrated on game development concepts and tools and it was open to other students if they were interested. The regular labs concentrated on writing larger object oriented programs.

One TA gave most of the instructions in the first 20-minute introductory session, followed by students' hands-on programming sessions monitored by both TAs. Detailed instructions are provided, which guide the students through their 90 minutes' programming practice. Considering that the students are freshmen most of whom do not have any experience in programming, the first 2 lab documents were designed to help them acquire basic knowledge of compiling and debugging C++ programs with Microsoft Visual Studio 2005 and on the UNIX system. Lab 3 and 4 provides them with exercises to practice their acquired knowledge in C++ programming through designing and developing small programs. In the remaining labs, most of the content and time is devoted to object oriented development. The games group spends their time on learning the GGE game engine and components of games. They develop a game as part of their final project requirement in addition to the normal class final project. The CS students who participated in the games group had two final projects for the semester.

Each game lab handout is set up with a learning goal, e.g., to create a window frame and resize it. The handout also explains concepts associated with game development, such as rendering, events and event handling, state machines, and so on. Then the handout provides detailed steps in creating simple toy-like games using the GGE package. One TA demonstrates the first few steps on the instructor desktop to start the students on the right track and get them to the same starting point. Then, the students are asked to use the code from the document, inserting that code at the appropriate places to modify the program and make it do what is required.

To reduce students' level of frustration when encountering problems during test-running their projects, they were encouraged to raise their hand whenever they had questions, and TAs would approach them to provide help with debugging the program and explanation as what caused the problem. This feedback-corrective procedure helped students to cope with the problems and provided us with an opportunity to modify and update our lab handout to correct confusing statements or instructions. This approach helped reduce the student's frustration due to the failure of building and running projects.

Gradually the students mastered the techniques of composing code using GGE. Then students were challenged by asking them to do advanced lab exercises to perform problem-solving tasks. These exercises were closely related to what they were taught. For example, in a game where two paddles can be moved through keyboard controls, the code for moving the right paddle was explained and given to them previously and the lab exercise would ask them to add necessary code to make the left paddle do the same thing.

By the 11th week, around 50% of students mastered the techniques of composing a multi-frame window which has navigating buttons on each frame. In the 12th week, all the students can program this kind of application. Meanwhile, half of them were able follow the instructions to complete a simple game and do lab assignments without the TAs' help. All the students in both groups were given the tools they needed to practice at home on their own computers to improve their skills.

ASSESSMENT OF THE CHANGES

We conducted an exit survey on the last week of class to measure the level of satisfaction and also the students' level of knowledge after completing the course. During the semester, informal discussions with students provided feedback to make minor adjustments to the process. The table below shows complete grade based comparison. Based on the compiled data, the results showed grades' of games group had a higher distribution of grades. There is a 28% increase on the number of students who received an A for the course in the Fall 08 and 49% increase for the Spring 09.

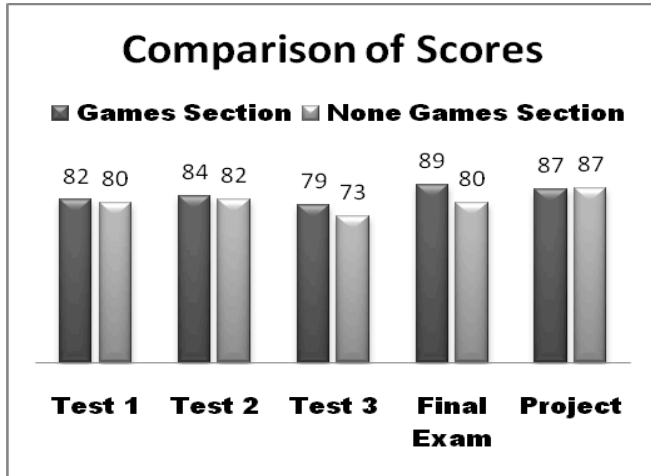
Fall 2008 Results

Grades	Games Group	Regular Group	Difference Games / Regular
A	47%	19%	28%
A-	13%	22%	-9%
B+	8%	14%	-7%
B	11%	10%	1%
B-	3%	7%	-5%
C+	5%	8%	-3%
C	0%	0%	0%
C-	3%	2%	0%
D+	3%	1%	1%
D	0%	1%	-1%
D-	0%	0%	0%
F	0%	7%	-7%

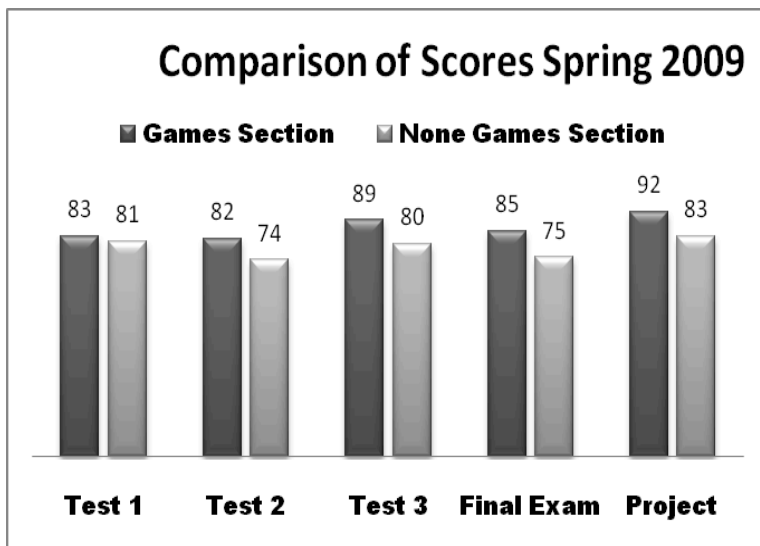
Spring 2009 Results

Grades	Games Group	Regular Group	Difference Games / Regular
A	64%	15%	49%
A-	12%	16%	-4%
B+	0%	9%	-9%
B	18%	3%	-15%
B-	6%	9%	-3%
C+	0%	5%	-5%
C	0%	9%	-9%
C-	0%	5%	-5%

Fall 2008 Results



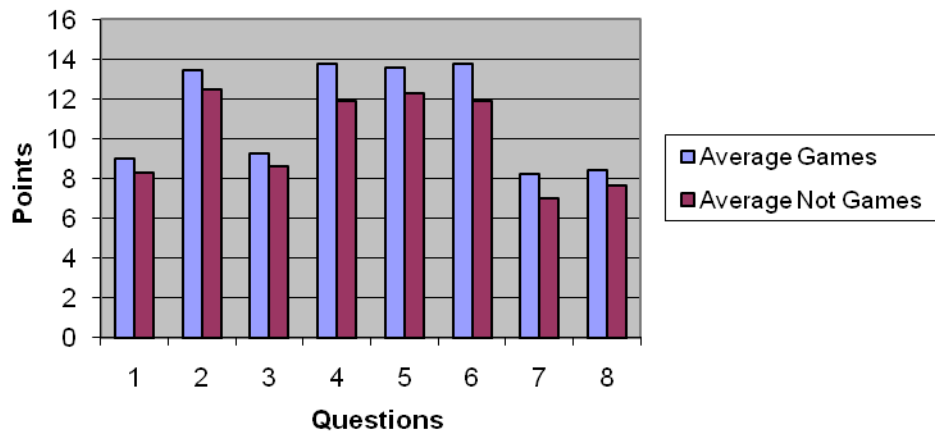
Sprint 2009 Results



We compared and evaluate the performance of students on each question of the final exam. The results show that students who participated in the games lab had higher scores on each of the questions on the final exam.

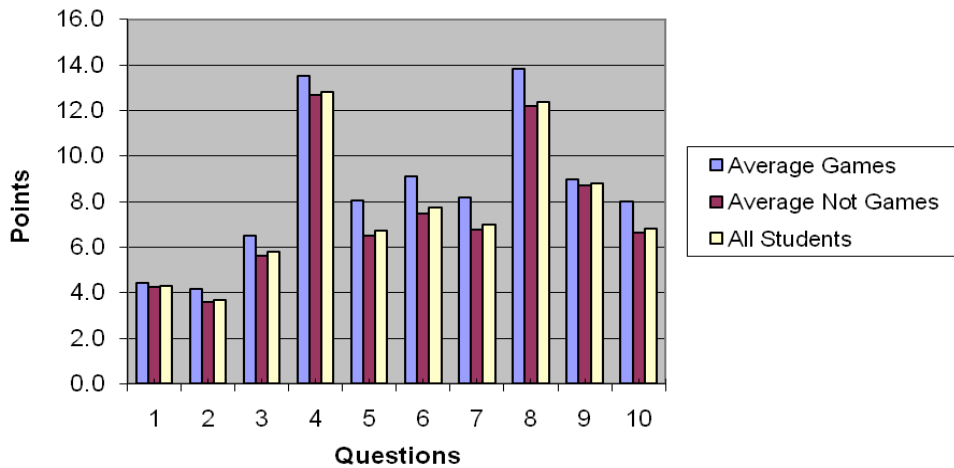
Fall 2008 Results

Final Exam Score Distribution



Spring 2009 Results

Final Exam Score Distribution Per Question



On the course final project games students also outperformed the other students by 10 points.

To have a true sense of effectiveness of the changes, we need to repeat the process for at least 1 more year and track the students who started this year as freshman to see if they perform the same in their later years at university.

SURVEY:

Exit survey results:

Gender?

Female 13 %
Male 87 %

What is your year at school?

Freshman 74 %
Sophomore 13 %
Junior 9 %
Senior 4 %

How did having a gaming component GGE in the lab help you better understand Object Oriented Programming concepts (using and designing classes)? (1 did not help , ..., 5 very Helpful)

Answers	Percent Answered
1	22 %
2	22 %
3	22 %
4	26 %
5	7 %

How well do you understand GGE game engine? (1 not at all,....., 5 very well)

1	13 %
2	35 %
3	13 %
4	30 %
5	9 %

How did having a gaming component GGE in the lab help you better understand programming concepts? (1 did not help , ..., 5 very Helpful)

1	17 %
2	22 %
3	30 %
4	22 %
5	9%

Did you finish the Game project?

Did not try	39%
Tried but could not finish it.	22%
Partially worked, but not completely.	4%
Completed the game and worked correctly.	35%

How did the Games lab meet your expectations (1 is the lowest and 5 is the highest)?

1	13 %
2	26 %
3	39 %
4	18 %
5	4 %

In the comment section of the survey:

1. Most students were very excited about the game's component.
2. Many students complained about the complexity of the gaming engine compare to their knowledge. To help elevate this problem, there will be more help sessions this semester to go over the engine in more details.
3. Most like to have more OOP and pointers in C++ before starting the game, we started the lab earlier and spend more time discussing this syntax related issues early on.
4. Better lab room and more meetings. We moved to a new lab this semester with brand new equipment, courtesy of the USC GamePipe Laboratory, and its director Mike Zyda.
5. Game engine complexity and difficulties. More time was allocated to explain and using the GGE with short exercises early on. A simplified tutorial was developed for the student use.
6. Difficulty understanding physics and math involved. Will attempt to get some guest lecturers to explain these concepts.

ACKNOWLEDGEMENTS

A special thanks to Microsoft Research that provided the necessary funding and support to implement the changes to CSci101. Mr. John Nordlinger from MSR was instrumental to provide the financial and training support for this project.

Massoud Ghyam, EdD

University of Southern California
Computer Science Department

Los Angeles, CA 92612

1-213-740-4515

mghyam@usc.edu